**Homework 7 : xv6 locking**

2017030519 홍유진

**1. Don't do this**

```
// Allocate a file structure.
struct file*
filealloc(void)
{
  struct file *f;

  acquire(&ftable.lock);
  acquire(&ftable.lock);
  for(f = ftable.file; f < ftable.file + NFILE; f++){
    if(f->ref == 0){
      f->ref = 1;
      release(&ftable.lock);
      return f;
    }
  }
  release(&ftable.lock);
  return 0;
}
```

file.c에 acquire을 추가하였다.

```
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
lapicid 1: panic: acquire
 8010440d 80100d92 8010507b 80104887 80105a49 8010570c 0 0 0 0
```

이미 holing되어있는 상태라서 panic이 발생한다. 즉 만약 xv6 커널이 이미 lock이 acquire되 있는데 또 lock을 걸면 deadlock이 된다. 그래서 커널은 panic을 나타낸다.

## 2. Interrupts in ide.c

```
void
iderw(struct buf *b)
{
  struct buf **pp;

  if(!holdingsleep(&b->lock))
    panic("iderw: buf not locked");
  if((b->flags & (B_VALID|B_DIRTY)) == B_VALID)
    panic("iderw: nothing to do");
  if(b->dev != 0 && !havedisk1)
    panic("iderw: ide disk 1 not present");

  acquire(&idelock);  //DOC:acquire-lock
  sti();
  // Append b to idequeue.
  b->qnext = 0;
  for(pp=&idequeue; *pp; pp=&(*pp)->qnext)  //DOC:insert-queue
    ;
  *pp = b;

  // Start disk if necessary.
  if(idequeue == b)
    idestart(b);

  // Wait for request to finish.
  while((b->flags & (B_VALID|B_DIRTY)) != B_VALID){
    sleep(b, &idelock);
  }

  cli();
  release(&idelock);
}
```

```
xv6...
cpu1: starting 1
cpu0: starting 0
lapicid 1: panic: acquire
 801043fd 80102063 80105a15 801056fc 80100183 801013c5 801014bf 801036c4 801056f
f 0
```

```
^C
Thread 1 received signal SIGINT, Interrupt.
0x801003e3 in panic (s=0x80107682 "sched locks") at console.c:121
121       panicked = 1; // freeze other CPU
(gdb) bt
#0  0x801003e3 in panic (s=0x80107682 "sched locks") at console.c:121
#1  0x80103bc1 in sched () at proc.c:378
#2  0x80103d32 in yield () at proc.c:394
#3  0x80105913 in trap (tf=0x8dfffec0) at trap.c:171
#4  0x801056fc in alltraps () at trapasm.S:20
#5  0x8dfffec0 in ?? ()
#6  0x801021ba in iderw (b=0x8010b5f4 <bcache+52>) at ide.c:159
#7  0x80100183 in bread (dev=1, blockno=1) at bio.c:103
#8  0x801013c5 in readsb (dev=1, sb=0x801109c0 <sb>) at fs.c:36
#9  0x801014bf in iinit (dev=1) at fs.c:182
#10 0x801036c4 in forkret () at proc.c:412
#11 0x801056ff in alltraps () at trapasm.S:21
(gdb)
80103bc1 80103d32 80105913 801056fc 801021ba 80100183 801013c5 801014bf 801036
```

: 함수로 들어갈 때 하나의 lock만 잡혀야 한다. 하지만 실제로 커널은 interrupt를 허용함으로써 두개의 lock을 hold할 수 있다. Idrew()에서 interrupt를 허용하면서 trap이 발생해 trap.c에서 yield를 호출한다. 이때 yeild에서 ptable.lock을 acquire한다. 이때 idrew에서 이미 acquire을 했기 때문에 lock이 두번 잡히게 된다.  따라서 yield()에서 sched()를 호출했을 때 ncli가 한 개가 아니므로 에러가 발생한다.

### 3. interrupts in file.c

: sleep()함수처럼 filealloc()엔 ptable.lock을 획득하는 방법이 없기 때문이다.

### 4. xv6 lock implementation

: 만약 커널이 state를 resetting하기전에 lock을 released하면 다른 프로세서는 inconsistent한 state에서 이것을 acquire할것이다. 왜냐하면 lk->cpu와 lk->pcs[] fields는 새로운 aqcuirer와 일치하지 않기 때문이다.